A Conceptual Model for Remote Data Acquisition Systems*

Txomin Nieva, Alain Wegmann

Institute for computer Communications and Applications (ICA),
Communication Systems Department (DSC),
Swiss Federal Institute of Technology (EPFL),
CH-1015 Lausanne, Switzerland

Abstract. Data Acquisition Systems (DAS) are the basis for building monitoring tools that enable the supervision of local and remote systems. DASs are complex systems. It is difficult for developers to compare proprietary generic DAS products and/or standards, and the design of a specific DAS is costly. In this paper we propose a conceptual model of a generic DAS. This model gives DAS developers an abstraction of DASs; it enables them to compare existing products and standards; and it provides the DAS developers that aim to develop a specific DAS with a starting point for the design of a specific DAS. We have found that a conceptual model of a generic system has many advantages. We propose patterns and techniques that are useful for the development of conceptual models of generic systems.

Keywords: Information System Engineering; Conceptual Modeling; Data Acquisition Systems; Remote Monitoring Systems; Condition Monitoring; Embedded Systems

1. Introduction

In the last few years, companies from many business areas have become increasingly interested in maintenance and asset management. Maintenance improves the reliability and availability of equipment and therefore the quality of service (QoS), which managers have found provides substantial benefits. Maintenance management however makes up anywhere from 15 to 40% of total product cost (Wireman, 1994). Consequently, improving maintenance management can also represent a substantial benefit to companies. Traditionally, there are two major maintenance approaches: Corrective Maintenance and Preventive/Predictive Maintenance (PPM). Corrective Maintenance focuses on efficiently repairing or replacing equipment after the occurrence of a failure. Corrective Maintenance aims to increase the maintainability of equipment by improving the speed of repair, or return to service, after a failure. PPM focuses on keeping equipment in good condition in order to minimize failures; repairing components before they fail. PPM aims to increase the reliability of equipment by reducing the frequency of failures. Substantial benefits can also be obtained by the intensive use of Asset Management Systems (AMS). Asset management is a task complementary to maintenance. It provides support for the planning and operation phases. Similar to maintenance tasks, in AMSs access to utility data source is essential.

A management technique that can be applied for improving maintenance and asset management is the on-line supervision of the health of the equipment, which is usually known as condition monitoring.

---

* Technical Report N° DSC/2001/030

Condition monitoring, applied to maintenance tasks, provides necessary data in order to schedule preventive maintenance and to predict failures before they happen. Condition monitoring is based on direct monitoring of the state of equipment to estimate its Mean Time To Failure (MTTF). AMSs will propose or update PPM plans based on the information provided by the condition monitoring systems. To apply condition monitoring, the access to utility data source is essential. Remote monitoring systems have been developed in many business areas such as building (e.g., Olken et al., 1998), power engineering (e.g., Itschner et al., 1998) and transportation systems (e.g., Fabri et al., 1999) to provide condition-monitoring systems with information about the state of equipment. The kernel of any remote monitoring system is a data acquisition system (DAS), which enables the collection of relevant data. A DAS is a set of hardware and software resources that provides the means to obtain knowledge-level data of a system, provides the means to access operational-level data, converts knowledge-level and operational-level data to more useful system information and distributes this information to the user. There are many standards for DASs such as OLE for Process and Control (OPC) (OPC Foundation, 1997), Interchangeable Virtual Instrument (IVI) (IVI Foundation, 1997) and Open Data Acquisition Standard (ODAS) (ODAA, 1998), among others. Additionally, the Object Management Group (OMG) has recently issued a Data Acquisition from Industrial Systems (DAIS) Request For Proposal (RFP) (OMG, 1999). Based on DAS standards, there are many commercial generic DAS products that DAS developers can buy and customize for their specific DAS application. DAS developers have to choose between buying a commercial DAS product and customizing it for their specific requirements or designing from scratch a specific DAS. However, DASs are complex systems. It is difficult for DAS developers to understand DAS standards and/or generic DAS products. As each standard or product uses a different idiom it is also difficult for DAS developers to compare them. Additionally, the development of a specific DAS from scratch is a difficult task that requires high development costs.

In this paper we propose a conceptual model of a generic DAS. This model gives DAS developers an abstraction of DASs; it enables them to compare existing products and standards; and it provides the DAS developers that aim to develop a specific DAS with a starting point for the design of a specific DAS. We have found that a conceptual model of a generic system has many advantages. We propose some patterns and techniques to develop conceptual models of generic systems. Additionally, our generic DAS model provides a case study of conceptual modeling of generic systems that demonstrates, by means of an industrial example, the advantages of conceptual modeling for the specification of generic systems.

This paper is organized as follows: In section 2, we explain the methodology we used to obtain a generic DAS conceptual model. In section 3, we present the generic DAS conceptual model. In section 4, we discuss key issues about the development of the generic DAS conceptual model. In section 5, we explain the applications of a conceptual model of a generic system. Finally, in section 6, we draw conclusions from the actual work.

2. Methodology

A conceptual model is a formal description of a system, from the object perspective, that shows the relevant concepts and relationships that make up this system. Using a conceptual model of a system makes it easier to understand the system, because the model only focuses on the main aspects of the system by hiding low-level details that render it difficult to understand. Boman et al. (1997) noted that:

"An effective approach to analyzing and understanding a complex phenomenon is to create a model of it. By a model is meant a simple and familiar structure or mechanism that can be used to interpret some part of reality. A model is always easier to study than the phenomenon it models, because it captures just a few of the aspects of the phenomenon".

The conceptual model presented in this paper is the result of an iterative process consisting of the object-oriented analysis, specification, implementation and deployment of a DAS for railway equipment (Fabri et al., 1999, Nieva, 1999). During this process, we used our own variation, which puts emphasis on role modeling, of the Catalysis (D'Souza and Wills, 1999) development process based on Unified Modeling Language (UML) (Rumbaugh et al., 1999). The concepts of the resulting model were generalized to be used in different domains other than transportation systems. Finally, this model was

compared to and enhanced with many DAS standards and the OMG's DAIS RFP. Our generic DAS conceptual model is also inspired by several software patterns. Appleton, B. (1997) defined software patterns as:

"Patterns for software development are a literary form of software engineering problem-solving discipline that has its roots in a design movement of the same name in contemporary architecture, literate programming, and the documentation of best practices and lessons learned in all vocations".

3. A Generic DAS Conceptual Model: the Model

In this section we present our generic DAS conceptual model. We show the main concepts that make up any DAS and their relationships. For a better understanding of the model, we group the concepts into five main packages. These packages and their inter-dependencies are shown in Figure 1.

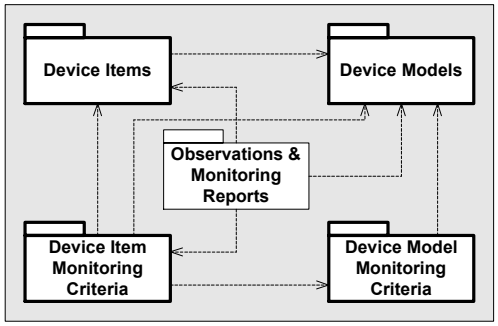

Figure 1 Data Acquisition Main Packages

- **Device Models**. This package groups all the concepts regarding device models. A device model represents a model that characterizes a set of device items.

- **Device Items**. This package groups all the concepts regarding device items. A device item represents a real world device that satisfies a device model.

- **Device Model Monitoring Criteria**. This package groups all the concepts regarding the definition of criteria, for generating monitoring reports, predefined in a device model and common for all the corresponding device items.

- **Device Item Monitoring Criteria**. This package groups all the concepts that allow for the definition of criteria, for generating monitoring reports, specific for a device item.

- **Observations & Reports**. This package groups all the concepts regarding observations and reports taken on a system. Observations are classified as quantitative (measurements) or qualitative (category observations), according to the measurements and observations analysis pattern described by Fowler (1997). Monitoring reports are classified as reports that record a change in the composition of the system (composition reports), reports that record a snapshot of the system at a specific time (status reports), and reports that indicate a certain state of the system (event reports).

In the following sub-sections we describe each of these packages. In the models, we distinguish between *operational-level* and *knowledge-level* concepts. We adopt this idea from Fowler (1997). At the *operational-level* the model records the day-to-day events of the domain, whereas at the *knowledge-level* the model records the general rules that govern this structure. We represent *knowledge-level* concepts by using a box with a thick border, and a box with a thin border represents *operational-level* concepts.

3.1 Device Models

The *Device Models* model is shown in Figure 2. A device model is an instance of Device Model that represents a model, created in the design process, that characterizes a set of real world devices. A device model can be composed of many device models, each of them implementing a specific function on the parent device model. We used our own variation of the *Composite* pattern, named *Model Composite* (further discussed in section 4.3), to manage the composition of device models. A device model defines many measurement points. An instance of *Measurement Point* defines a measurement point associated with a phenomenon type and a measurement type. An instance of *Phenomenon Type* represents something that can be quantitatively (e.g. `temperature`), or qualitatively (e.g. `door_status`), observed. A phenomenon type defines the units on which observations of this phenomenon type are expressed. We adopt the convention of using standard SI (metric) units, as proposed by Olken, F. et al. (1998), for phenomenon types. A phenomenon type also specifies the range within which a value is qualified as "normal". Eventually, a phenomenon type records the set of potential qualitative values that a measurement of such a phenomenon type can take (e.g. `temperature_low`, `temperature_medium`, and `temperature_high`). Each of these values is an instance of *Phenomenon*. Phenomenon records the range of quantitative observations of a phenomenon type that corresponds to a qualitative observation. This enables the automatic recording of an occurrence of this phenomenon upon a quantitative observation, of the corresponding phenomenon type, with a value within the range of the phenomenon. An instance of *Measurement Type* is associated with a measurement point to give some semantic information about the measurements taken at this measurement point.

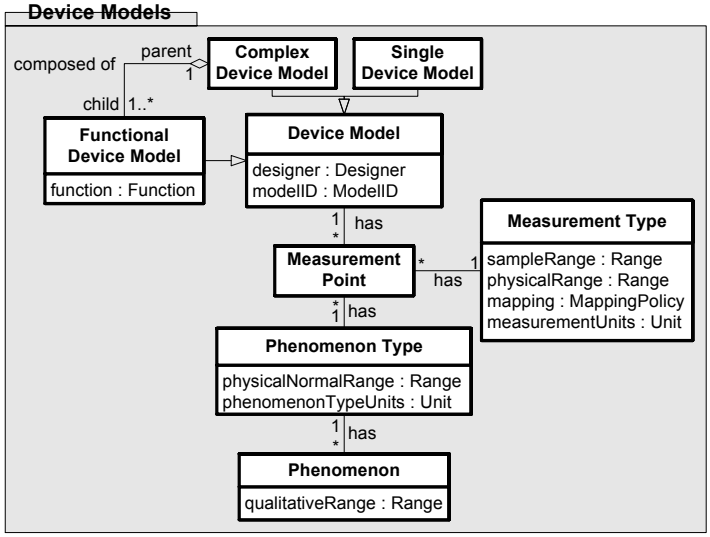

Figure 2 Device Models

3.2 Device Items

The *Device Items* model is shown in Figure 3. A device item is an instance of Device Item that represents a real world device created in the manufacturing process. A device item is characterized by a device model. As specified by its device model, a device item can be composed of many device items, which are also characterized by the associated device models. We organize device items using the *Composite* pattern. The association class *Device Address* allows us to record the address within a complex device item where a child device item, characterized by an instance of *Functional Device Model*, is installed. Each device item defines many measurement addresses. An instance of *Measurement Address* defines the actual location in a device item associated with a measurement point, where observations of a phenomenon type are taken.

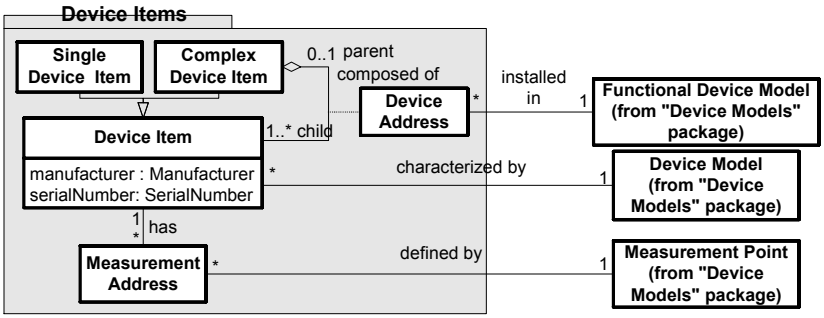

Figure 3 Device Items

3.3 Device Model Monitoring Criteria

The *Device Model Monitoring Criteria* model is shown in Figure 4. The ability to define reports, with a consistent status, of a set of data is one of the requirements of any DAS. The OMG's DAIS RFP (OMG, 1999) refers to this as *Dataset*, and OPC (OPC Foundation, 1997, OPC Foundation, 1999) as *OPC Group*. Our approach for defining criteria for recording monitoring reports is inspired by Mansouri-Samani and Sloman (1994). In the design process of a device model, a designer can define certain criteria specific to this device model and common to all the device items characterized by this device model. *Device Model Trigger Condition* enables the definition of conditions in order to automatically trigger the recording of reports at a specific time or when the system is in a certain state. We distinguish three kinds of monitoring criteria: *Device Model Composition Monitoring Criteria* enables the recording of changes on the composition of the system. A device model composition monitoring criteria is associated with a set of device models; *Device Model Status Monitoring Criteria* enables the recording of a snapshot of the system at a specific time. Status monitoring criteria are associated with a set of measurement points; and *Device Model Event Monitoring Criteria* enables the recording of a certain state of the system.

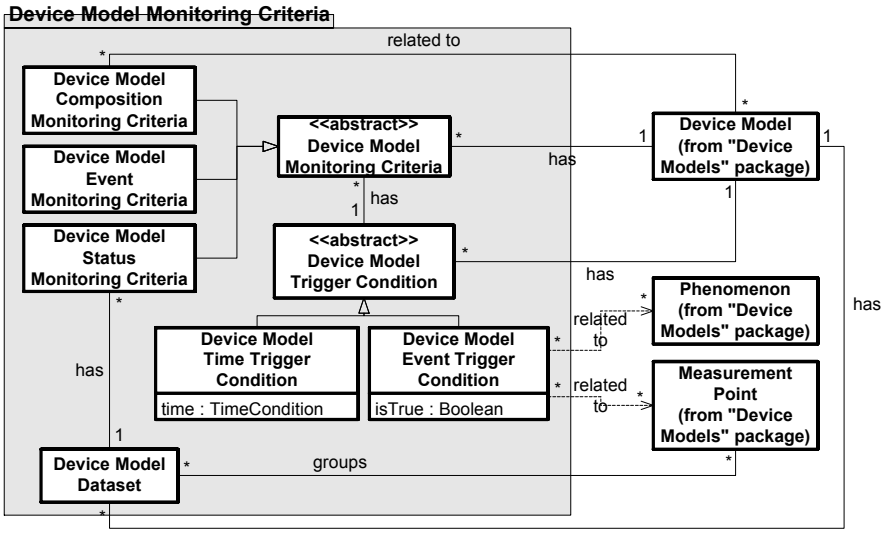

Figure 4 Device Model Monitoring Criteria

3.4 Device Item Monitoring Criteria

The *Device Item Monitoring Criteria* model is shown in Figure 5. In the installation and maintenance phases administrators administer datasets, trigger conditions and monitoring criteria for device items. The

DAS system will record monitoring reports corresponding to device item monitoring criteria. Device item datasets, trigger conditions and monitoring criteria can be *predefined*, meaning that there are already defined in the corresponding device model, or *custom*, meaning that there are defined in the device item. A custom monitoring criteria may use any combination of custom or predefined datasets and trigger conditions. Device item monitoring criteria can be *public*, meaning that any supervisor of the system can access monitoring reports of such criteria, or *private*, meaning that only the creator of the monitoring criteria is allowed to access monitoring reports corresponding to such criteria.

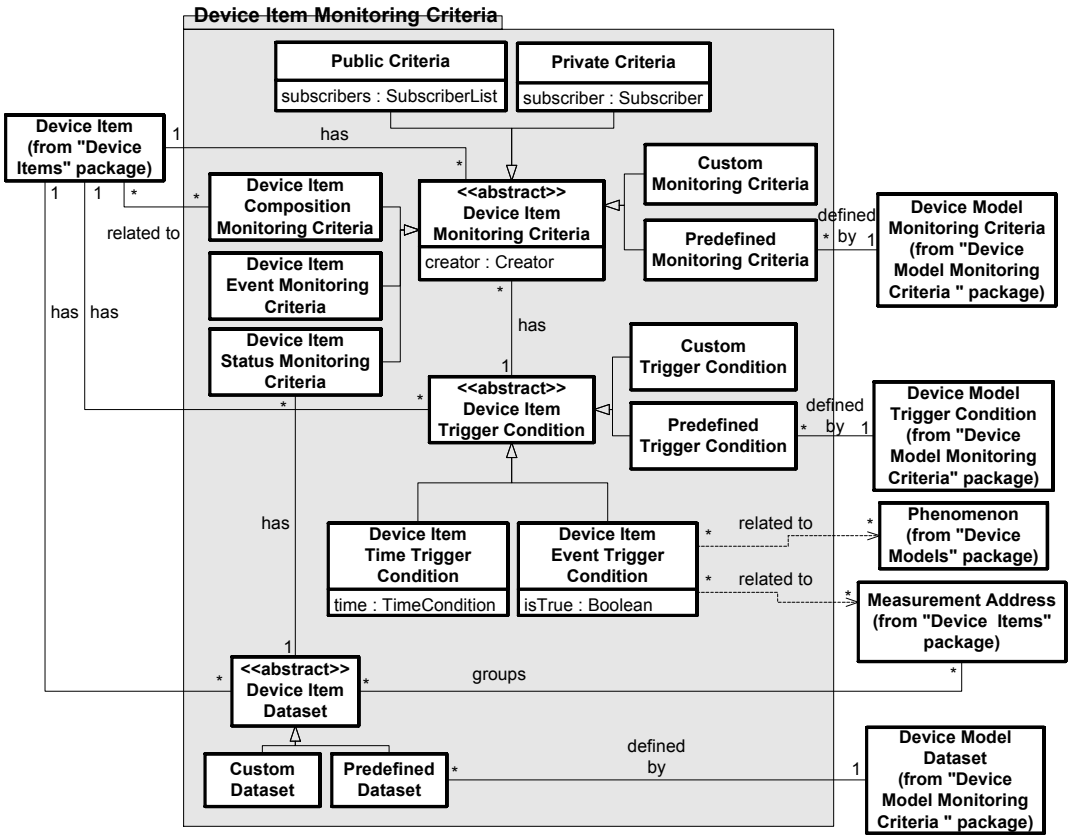

Figure 5 Device Item Monitoring Criteria

3.5 Observations & Monitoring Reports

The *Observation and Monitoring Reports* model is shown in Figure 6. In this package we present concepts that allow us to record observations and monitoring reports taken on a device item. Our observation model is inspired by the *Observations and Measurements* analysis pattern described by Fowler (1997). In a measurement address we can record many observations with different timestamps. *Observation* is an abstract concept that represents both quantitative and qualitative observations. An observation records a timestamp corresponding to the time an observation was taken. *Measurement* represents quantitative observations. A measurement records the physical value corresponding to the measurement, which is represented by the *value* attribute. A measurement is associated with a phenomenon type, and a phenomenon type can have many measurements. A *Category Observation* represents a qualitative observation. A category observation is associated with a phenomenon, and a phenomenon can have many category observations. Sometimes recording that a phenomenon is absent is as important as recording its presence. The *isPresent* Boolean attribute of category observation is added to enable recording the absence or presence of a phenomenon. *Monitoring Report* enables the recording of an occurrence of fulfilled monitoring criteria. A monitoring report is always associated with monitoring criteria of a device item. A monitoring report is also associated with the observations that generate the monitoring report.

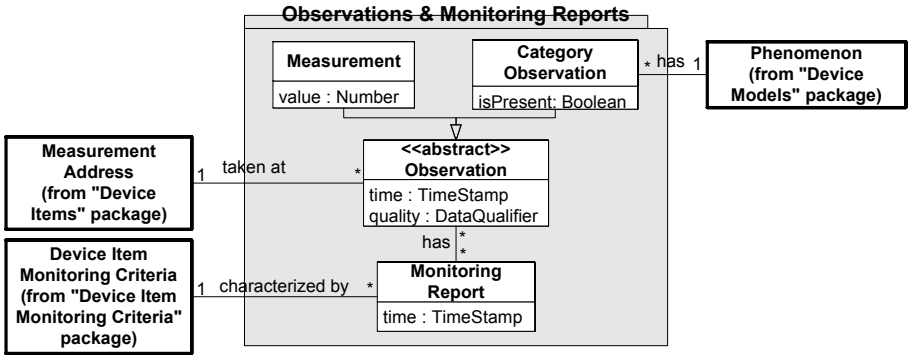

Figure 6 Observations & Monitoring Reports

3.6 Detailed Concepts

In this section we explain some detailed concepts of our generic DAS conceptual model. These concepts are: Measurement Type, Mapping Policy, Time Condition, Device Model Event Trigger Condition, Device Item Event Trigger Condition, Timestamp and Data Qualifier.

Measurement Type. The *Measurement Type* model is shown in Figure 7. A measurement type defines the permissible ranges of sampled and physical values, a mapping policy between sampled and physical values, and the units of the measurement. This information could have been embedded in a phenomenon type, but measurement type allows us to reuse the same information for several phenomenon types. The measurement type information depends on how measurements are actually measured in a measurement point. As a consequence it may happen that the measurement units are not the same as the phenomenon type units, being then necessary to transform the physical value in measurement type units to the physical value in phenomenon type units, which will be recorded in the system.

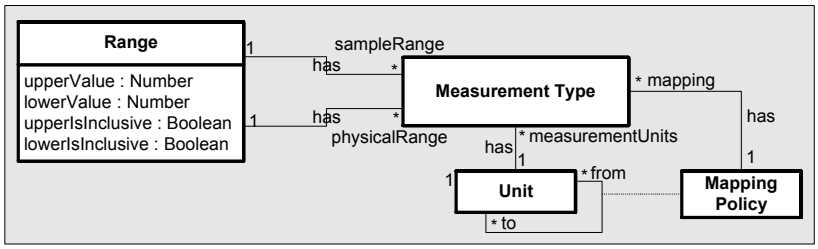

Figure 7 Measurement Type

Mapping Policy. The *Mapping Policy* model is shown in Figure 8. An instance of *Mapping Policy* defines the conversion between two numerical values. *Linear Mapping Policy* represents a mapping policy with a linear function (`y=Ax+B`); where `x` corresponds to the original value and `y` to the calculated value. *Function Mapping Policy* represents a mapping policy with a more complex function (`y=f(x)`). *Function Mapping Policy* seems to be a good scenario for applying mobile code (Carzaniga et al. 1997). In this way, device designers could easily upload the code corresponding to this function.

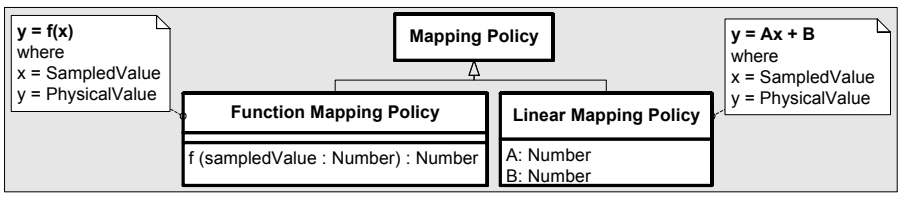

Figure 8 Mapping Policy

Time Condition. The *Time Condition* model is shown in Figure 9. *Time Condition* enables the definition of periodical or scheduled time conditions to record reports. *Period* enables the definition of a time condition as a period of time in milliseconds. *Schedule* enables the definition of a schedule when a report will be generated. A schedule is composed of a recurrence time, a recurrence pattern and a recurrence range. *Recurrence Time* enables the definition of a 24-hour period when a report will be generated. *Recurrence Pattern* enables the definition of many ways to record a time pattern for occurrences of a report: *Daily*, *Weekly*, *Monthly* and *Yearly* allow us to define a report to be generated every certain number of days, every certain number of weeks on some specific days of a week, every certain number of months on a specific day of a month, and every certain number of years on a specific day of a month, respectively. Finally, *Recurrence Range* enables the definition of a beginning time (just a *Begin Date*) to start recording reports and an end time to stop recording reports. *End Time* enables the definition of an end time by a number of occurrences, by a specific end date or with no end (meaning that the report will be generated "forever").

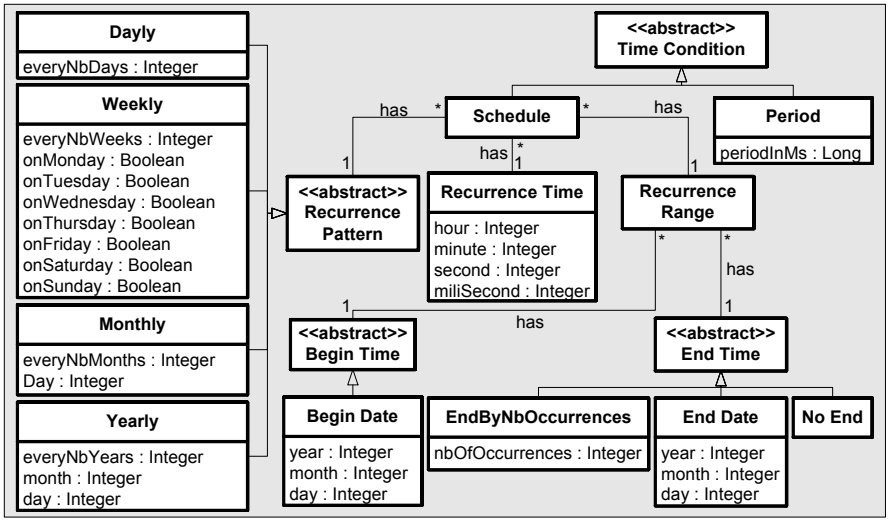

Figure 9 Time Condition

Device Model Event Trigger Condition. The *Device Model Event Trigger Condition* model is shown in Figure 10.

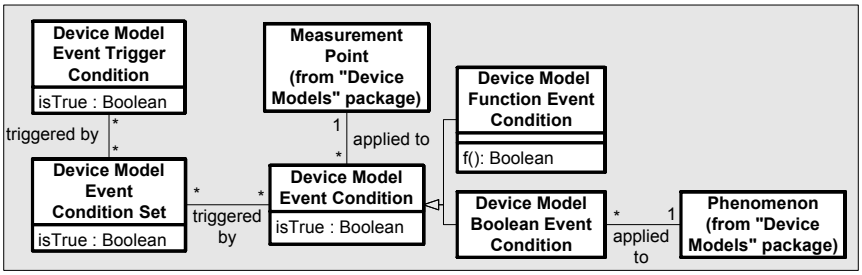

Figure 10 Device Model Event Trigger Condition

The *Device Model Event Trigger Condition* model enables the definition of conditions, common to all the device items characterized by a device model, to trigger an event. In order to explain device model event trigger conditions, we make use of a Boolean algebraic notation[1]. *Device Model Event Condition*

1 " . " corresponds to the "AND" logical operator; " + " corresponds to the "OR" logical operator; and " ′ " corresponds to the "NOT" logical operator

enables the recording of `X=A` and `X=A'`; where `A` means that a certain condition has been satisfied. There are two kinds of *Device Model Event Conditions*. *Device Model Boolean Event Condition* enables the recording of a condition that is satisfied when a certain phenomenon has been observed in a measurement point. *Device Model Function Event Condition* enables the recording of a condition that is satisfied when the result of applying a certain function in a measurement point returns true. *Device Model Function Event Condition* seems to be a good scenario for applying mobile code (Carzaniga et al., 1997). In this way device designers and/or administrators could easily upload the actual code corresponding to the function to be satisfied. *Device Model Event Condition Set* enables the recording of conditions such as `X=A.B` and `X=(A.B)'`; where `A`, `B` are *Device Model Event Conditions*. *Device Model Event Trigger Condition* allows us to record trigger conditions such as `X=A+B` and `X=(A+B)'`; where `A`, `B` are *Device Model Event Condition Sets*. This enables the recording of any device model event trigger condition, because any device model event trigger condition can be expressed by means of an algebraic combination of device mode event conditions with the `AND` logical operator and an algebraic combination of device model event condition sets with the `OR` logical operator. A transformation of any algebraic expression into these terms is possible by applying one of *De Morgan's* laws[2].

Device Item Event Trigger Condition. The *Device Item Event Trigger Condition* model is shown in Figure 11. The *Device Item Event Trigger Condition* model enables the definition of conditions, specific to a device item, to trigger an event. The reasoning is analogous to the device model event trigger condition, but the difference is that the condition is applied to a specific measurement address of a device item rather than to a measurement point of a device model.

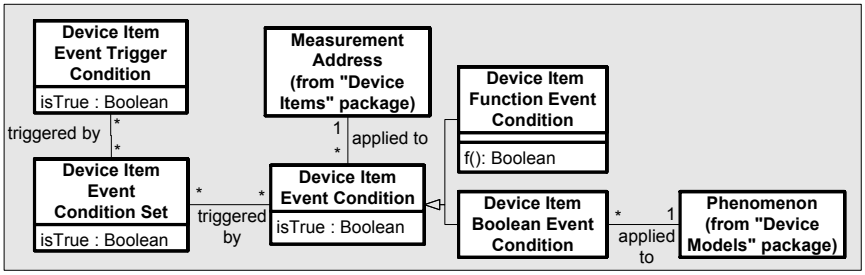

Figure 11 Device Item Event Trigger Condition

Timestamps. Recording the time an observation is taken is a key issue for enabling a subsequent analysis of observations. In order to avoid anomalies due to inconsistent time formats (e.g., because of different time zones), we adopted the convention of storing all timestamps using Universal Coordinated Time (UTC) format. This, further discussed by Olken et al. (1998), is a common practice in DASs.

Data Qualifiers. In DASs it is also a common practice to include a data qualifier (see Figure 12) with an observation. According to OMG (1999) a *Data Qualifier* includes information about the *Validity* (`valid`, `held` from a previous value, `suspect`, `not_valid` or `substituted` manually, the *Current Source* (`metered`, `calculated`, `entered`, or `estimated`) and the *Normal Value* (`normal` or `abnormal`) of an observation.

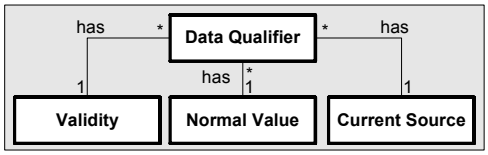

Figure 12 Data Qualifier

2 The two laws, known as *De Morgan*'s, are: `(A+B)'=A'.B'`; and `(A.B)'=A'+B'`

3.7 Complete Generic DAS Conceptual Model

The complete conceptual model of a generic DAS is shown in Figure 13.[3]

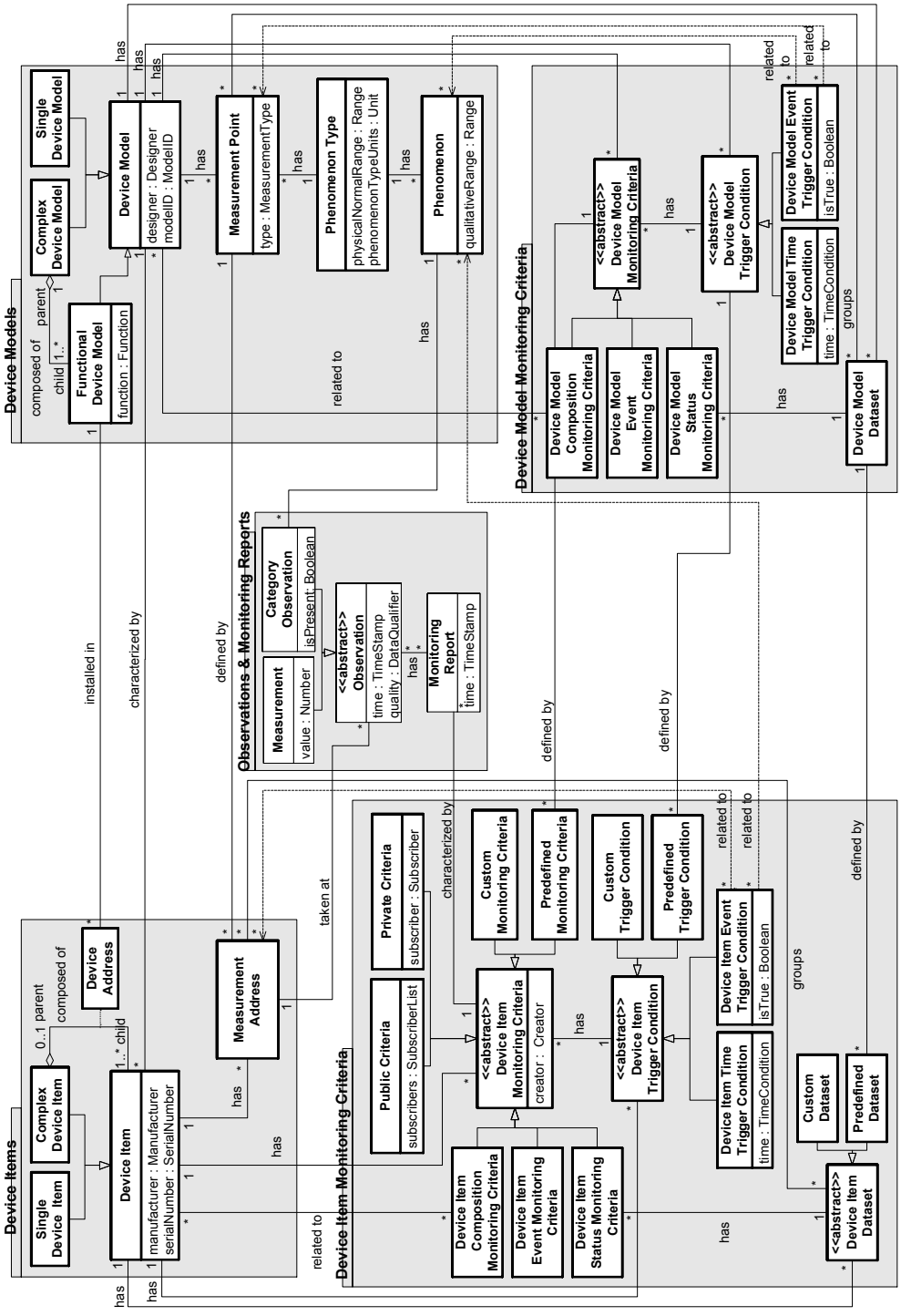

Figure 13 Generic DAS Conceptual Model

[3] To simplify the model we only show the main attributes of a concept and some concepts have been intentionally designed as attributes of higher-level concepts.

4. A Generic DAS Conceptual Model: Discussion

In this section we discuss key issues about the development of our generic DAS conceptual model. We discuss the relationship between device models and device items; we define how to assign a Global Unique Identifier (GUID) to device models and device items; we discuss the composition of device models and device items introducing a new pattern, called *Model Composite*, for the management of the composition of models; we explain how our conceptual model supports the notion of *Plug&Play*; and finally we explain the mapping between sampled and physical values.

4.1 Device Models vs. Device Items

We refer as *device* to a generic concept for any industrial system or sub-system. In our model, we represent a real world device as an instance of *Device Item*. We represent the type of a device, commonly known as its *model*, as an instance of *Device Model*. The relationships between instances of *Device Item* and instances of *Device Model* are shown in Figure 14. An instance of *Device Item* is always characterized by an instance of *Device Model*. An instance of *Device Model* characterizes a set of instances of *Device Item*.

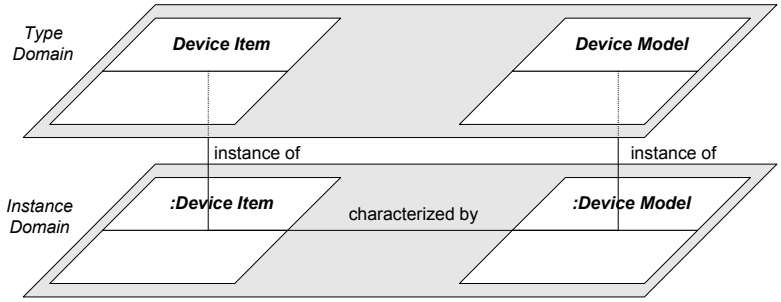

Figure 14 Device Item vs. Device Model

4.2 Naming Management

A Global Unique Identifier (GUID) must be assigned to each device model and device item. In this section we define how to assign a GUID to device models and device items.

Device Model Identifier. Device model designers are responsible for assigning a designer specific model identifier to their device models. This identifier, which we named *modelID*, enables the distinction between two different device models belonging to the same designer. A designer identifier, which we named *designerID*, enables the distinction between two different device model designers. As a result, a *deviceModelGUID* is obtained from the concatenation of *designerID* and *modelID*.

deviceModelGUID = designerID & *modelID*

Device Item Identifier. Device manufacturers are responsible for assigning a unique identifier, which is named *serialNumber*, to each device item. *serialNumber* uniquely identifies device items of the same device model manufactured by a manufacturer. In order to be able to globally identify a device item, it is necessary to include the *deviceModelGUID*. As a result, a *deviceItemGUID* is obtained from the concatenation of its corresponding *deviceModelGUID*, the *manufacturerID* and a *serialNumber*.

deviceItemGUID = deviceModelGUID & manufacturerID & serialNumber
deviceItemGUID = designerID & modelID & manufacturerID & serialNumber

4.3 Composition Management

An industrial system is usually composed of many *parts*, which can also be composed of many other *parts* in a *part-whole* hierarchy. In this section we discuss the management of the composition of device models and device items.

Device Items Composition Management. In DASs, tree structures allow us to efficiently define an industrial system. For example, an HVAC (heating, ventilation and air conditioning) system is composed of subsystems such as heating coil, cooling coil, supply fan, etc., that can be composed of other subsystems such as temperature sensors, ventilation sensors and so on. *Part-Whole* relationships are commonly used to model the construction of composite objects out of individual parts. *Part-Whole* relationship categories and their application in object-oriented analysis are further discussed by Motschnig-Pitrik and Kaasboll (1999). One way to represent *part-whole* relationships of systems is by using the *Composite* (Gamma et al., 1995) pattern. We used this pattern to organize device items.

Device Models Composition Management. An instance of *Device Item* is characterized by an instance of *Device Model*. As a result, there is an analogous relationship between pairs of device models and pairs of the corresponding device items. But, in the case of device models, the same device model may be used many times as part of the same complex device model. This impedes the use of the *Composite* pattern for the management of device models, as it would be not possible to distinguish between the different instances of the same *Device Model*.

Example: as shown in Figure 15, an instance of `VehicleModelA` is composed of two instances of `DoorModelB`. This is typically the case of vehicles with a `left door` and a `right door` of the same model. The problem is that there is no way to distinguish between the two instances of the same model `DoorModelB`, which are both part of `VehicleModelA`.

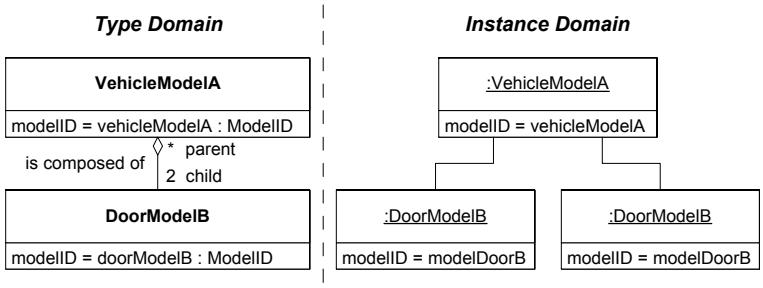

Figure 15 Example of Model Composition without Functional Model

An elegant manner to solve this problem is to use the *Model Composite* pattern. The *Model Composite* pattern, shown in Figure 16, is our own variation of the *Composite* pattern, to represent *part-whole* hierarchies of models.

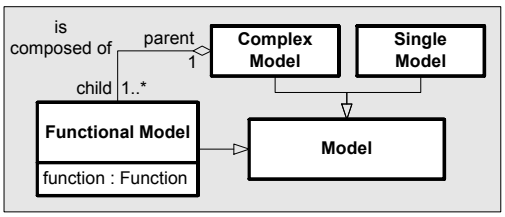

Figure 16 Model Composite Pattern

Model implements default behavior for a model. *Complex Model* defines behavior for a model that is composed of other models. *Functional Model* is a specialization of *Model* that represents a model that is part of a complex model. We called it *functional* because it implements a function within a complex

model. Using this pattern we may have many different instances of *Functional Model* that inherit from the same instance of *Model*, but each of them implements a different function on a complex model.

Example: as shown in Figure 17, a `VehicleModelA` is composed of a `VehicleModelALeftDoorDoorModelB`, implementing the function of `leftdoor` and a `VehicleModelBRightDoorDoorModelB`, implementing the function of `rightdoor`, which inherit both from `DoorModelB`. In this way we can distinguish between the two instances of the same model `DoorModelB`.

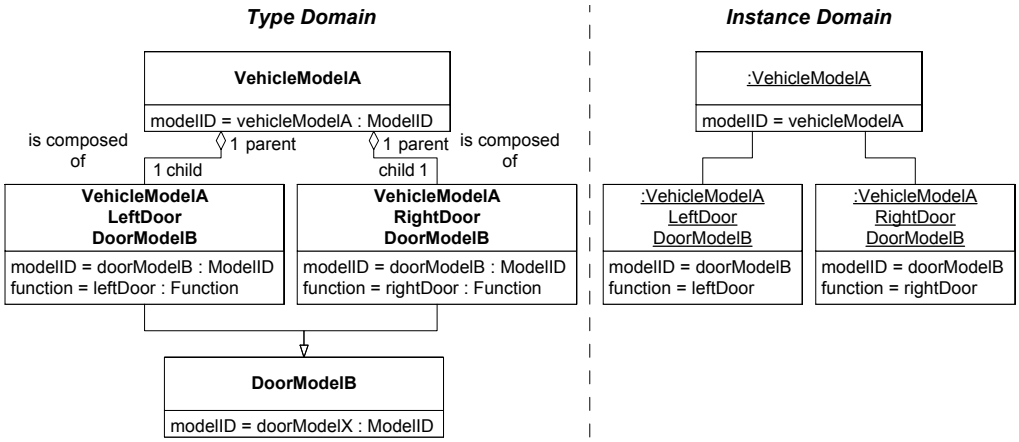

Figure 17 Example of Model Composition with Functional Model

4.4 Plug & Play

Plug&Play indicates that a system has the ability to automatically configure itself. The system must be able to detect changes in its composition to adapt its configuration to the new composition. One of the most known *Plug&Play* initiatives is Microsoft's Plug&Play (PnP) (MSDN, 1994), which is a framework architecture for PCs to enable the automatic configuration of expansion cards and other devices. Other initiatives, such as Universal Plug&Play (UPnP) (UPnP Forum, 2000) and Jini (Jini User Group, 2000), enable the *Plug&Play* of systems, or services, in a network.

A *Plug&Play* DAS would allow supervisors of devices to register to be notified when changes in the composition of the system happen. A *Plug&Play* DAS automatically detects changes in the composition of the system and notifies to interested supervisor of such changes. If the real world devices support the *Plug&Play* functionality, a *Plug&Play* DAS may subscribe itself in the devices to receive notifications when changes on the composition of such devices happen. Otherwise, a *Plug&Play* DAS may check periodically the composition of device items to detect eventual changes on their composition.

Our conceptual model supports the notion of *Plug&Play* through the concepts of *Device Model Composition Monitoring Criteria* and *Device Item Composition Monitoring Criteria*; these concepts allow for the definition of composition monitoring criteria, predefined in a device model or defined specifically in a device item, respectively. A composition monitoring criteria groups a set of devices. A supervisor may subscribe to receive a notification, by means of a monitoring report, when a change in the composition of, at least, one of the devices of such monitoring criteria happens. Our conceptual model defines the concepts that are necessary to enable the development of a *Plug&Play* DAS, but it does not force developers the use a specific technology. In an actual implementation, developers will chose the *Plug&Play* technology that fits better with their specific requirements.

4.5 Physical Values vs. Sampled Values

In DASs, it is very common for the value actually measured (we refer to this value as *sampled value*)

to not correspond to the *physical value*. We need a way to record a mapping policy that makes it possible to calculate the *physical value* from the *sampled value*. Eventually, the units in which a measurement is taken could not correspond to the measurement of the associated phenomenon type. This could happen because the measurement units depend on the sensor uses to acquire the measurement and not on the phenomenon type. Thus, we also need to record the units of the measurement in order to makes it possible to calculate the *physical value* in phenomenon type units from the physical value in measurement units. In order to record this information we introduced the concept of measurement type and mapping policy.

5. Application and Validation

In this section we explain the applications of a conceptual model of a generic system. The most direct application of a conceptual model of a generic system is for the writing of a RFP for a new standard. Another potential application of a conceptual model of a generic system is for the evaluation of existing systems, standards or RFP responses. We illustrate this by using our generic DAS conceptual model to compare different DAS standards. Finally, a conceptual model of a generic system can be applied in the development of a particular system. This will significantly reduce the development costs of a specific system. We illustrate this by means of an example of development of a DAS for railway equipment based on our generic DAS conceptual model.

5.1 Issuing/Replying a RFP

This is probably the most direct application of a conceptual model of a generic system. RFP issuers may use a conceptual model of a generic system as a guide to specify the static concepts that a proposal of standard in request for a new RFP must support and the functionalities that such a standard must deal with, creating and writing down a specific RFP. RFP repliers may use a conceptual model of a generic system to easier understand and analyze the requirements of this RFP. Additionally, RFP repliers can use a conceptual model of a generic system to describe their proposal of standard in request to this RFP. In this way a conceptual model of a generic system acts as an efficient communication mechanism between RFP issuers and RFP repliers, facilitating the creation, writing, understanding and replying of a RFP. A conceptual model of a generic system can be seen as an actor that actively collaborates in all these actions, as shown in Figure 18.

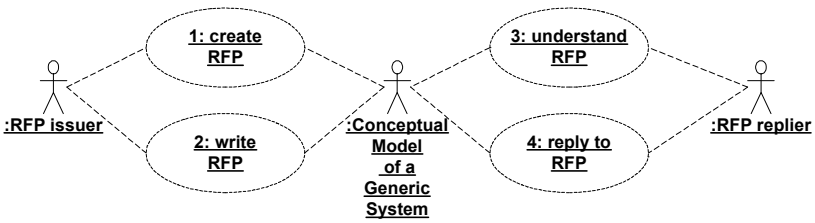

Figure 18 Issuing/Replying a RFP

5.2 Evaluation of Existing Systems or Proposals

Another potential application of a conceptual model of a generic system is the evaluation of existing systems or standards. Developers may use a conceptual model of a generic system to check the concepts that these systems or standards support. As an example, we used our generic DAS conceptual model to compare different DAS standards (OPC, IVI and ODAS).

OPC. The conceptual model with the concepts OPC supports is shown in Figure 19. OPC is a standard for process control in the automation field. In an *OPC Server* there are many *OPC Items*. An *OPC Item* is quite similar to a *Measurement Point*. It also provides some semantic information such as that corresponding to *Measurement Type* (data type, units, ranges and scales), *Phenomenon Type* and *Phenomenon* (implicit within the data type). As OPC does not support the concept of *model*, this semantic

information of an *OPC Item* is duplicated in all *OPC Items* corresponding to the same kind of *Measurement Point*. Additionally, it is not possible to define *Monitoring Criteria* predefined in a device model. We can say that OPC supports partially the notion of *composition* because an *OPC Server* can manage a flat list of devices. However, OPC does not deal with complex hierarchical structures of devices. OPC allows clients to define *custom trigger conditions* and *custom datasets*. *Datasets* are called *OPC Groups*, which group a set of *OPC Items* to be retrieved at the same time. However, OPC does not implement the *composition monitoring criteria* concept to enable the detection of changes on the composition of the system.

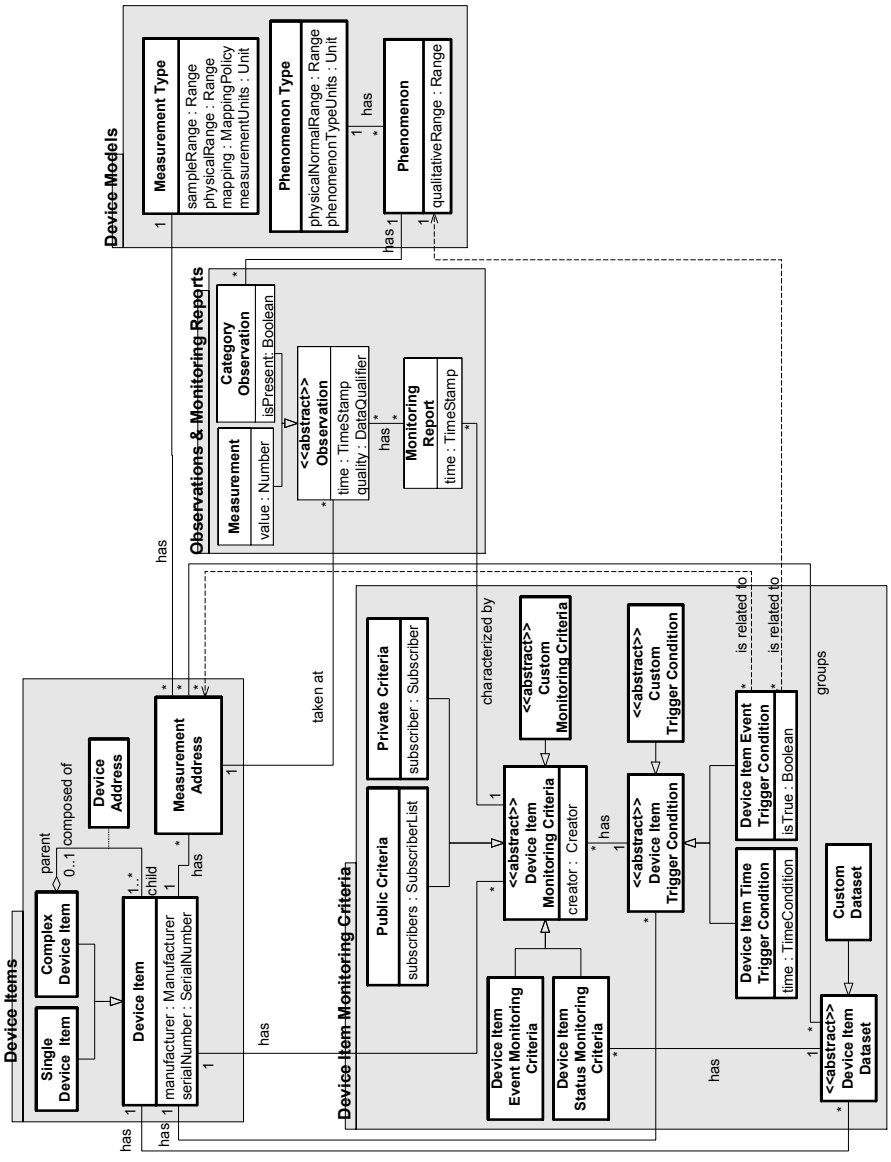

Figure 19 OPC Conceptual Model

IVI. The conceptual model with the concepts IVI supports is shown in Figure 20. IVI is a standard for instrumentation to enable the interchangeability among instruments of different vendors. In the IVI specification they refer to *instrument* or *assets* rather than devices. IVI defines many *instrument classes* such as *Oscilloscope* (IviScope), *Digital Multimeter* (IviDmm), *Function Generator* (IviFGen), *Switch* (IviSwtch), and *Power Supply* (IviPower) classes. An *IVI Class* can be considered a similar concept to a generic *Device Model*. IVI classes only deal with single devices. Additionally, IVI specifies the *IVI Measurement and Stimulus System* (IVI-MSS), which allows building complex *virtual instruments* composed of many real instruments. Thus, we can state that IVI supports somehow the *composition* of

devices through IVI-MSS. IVI is a standard defined to enable not only the interchangeability among instruments but also the *Plug&Play* of instruments. IVI is based on the VXIPlug&Play standard to enable a *Plug&Play* functionality. IVI classes specify *predefined trigger conditions* and *predefined datasets*. These trigger conditions may be enabled or disabled on a specific device item. Additionally, IVI MSS allows the definition of *custom trigger conditions* and *custom datasets*. Criteria defined in an IVI instrument class or IVI MSS system is always *public* to any IVI client, as IVI does not support the notion of *private criteria*.

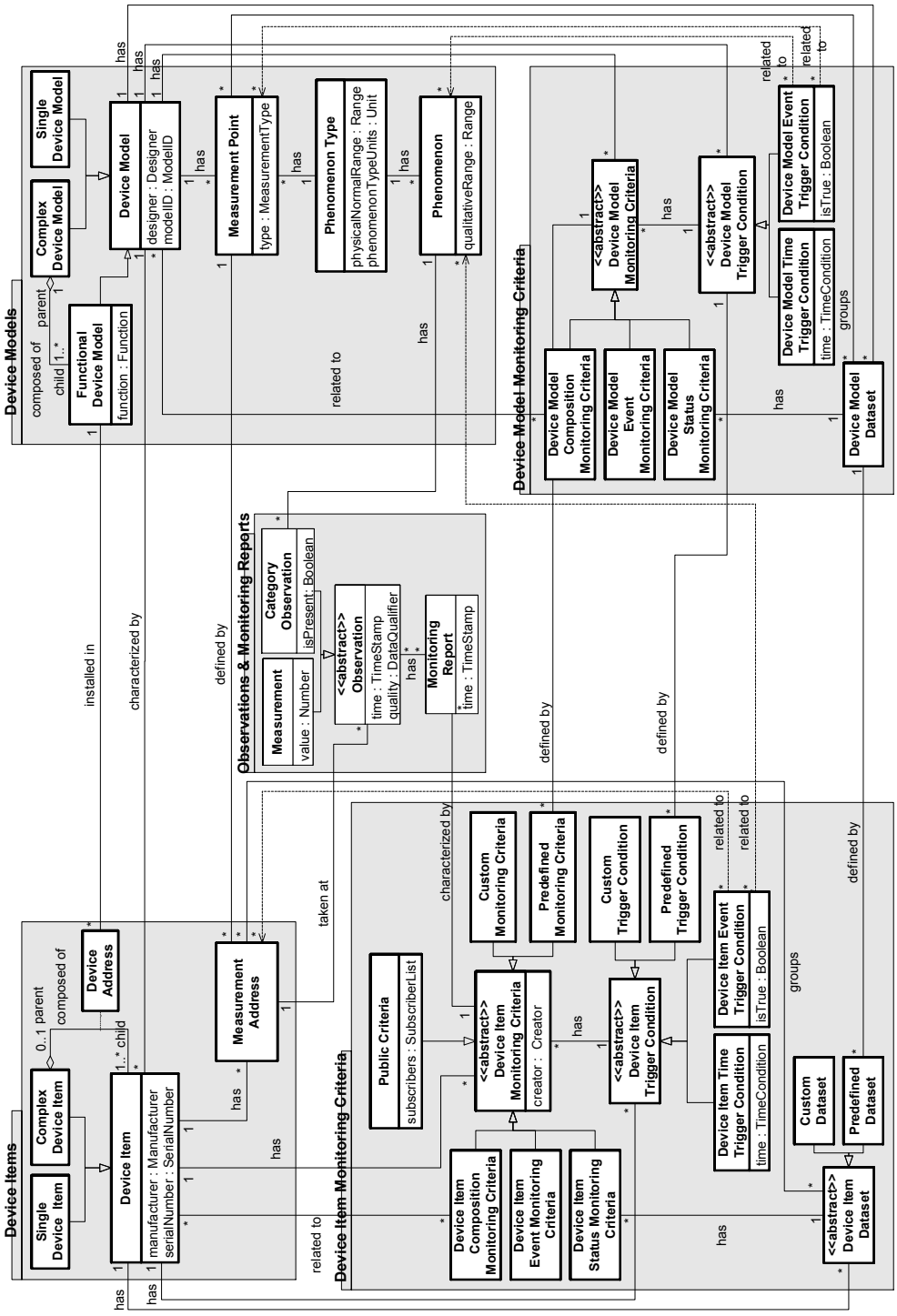

Figure 20 IVI Conceptual Model

ODAS. The conceptual model with the concepts ODAS supports is shown in Figure 21. ODAS is a standard for PC-based data acquisition systems. ODAS defines standard interfaces for *analog/digital inputs and outputs*. The scope of ODAS is single devices. ODAS does not support the notion of *model*. It does not support either the notion of *composition*. Only *custom datasets* and *custom event-based trigger conditions* may be defined by an ODAS client application. This allows clients to define *status monitoring criteria*. Criteria defined by a client is always *private* for this client, as ODAS does not support the notion of *public criteria*.

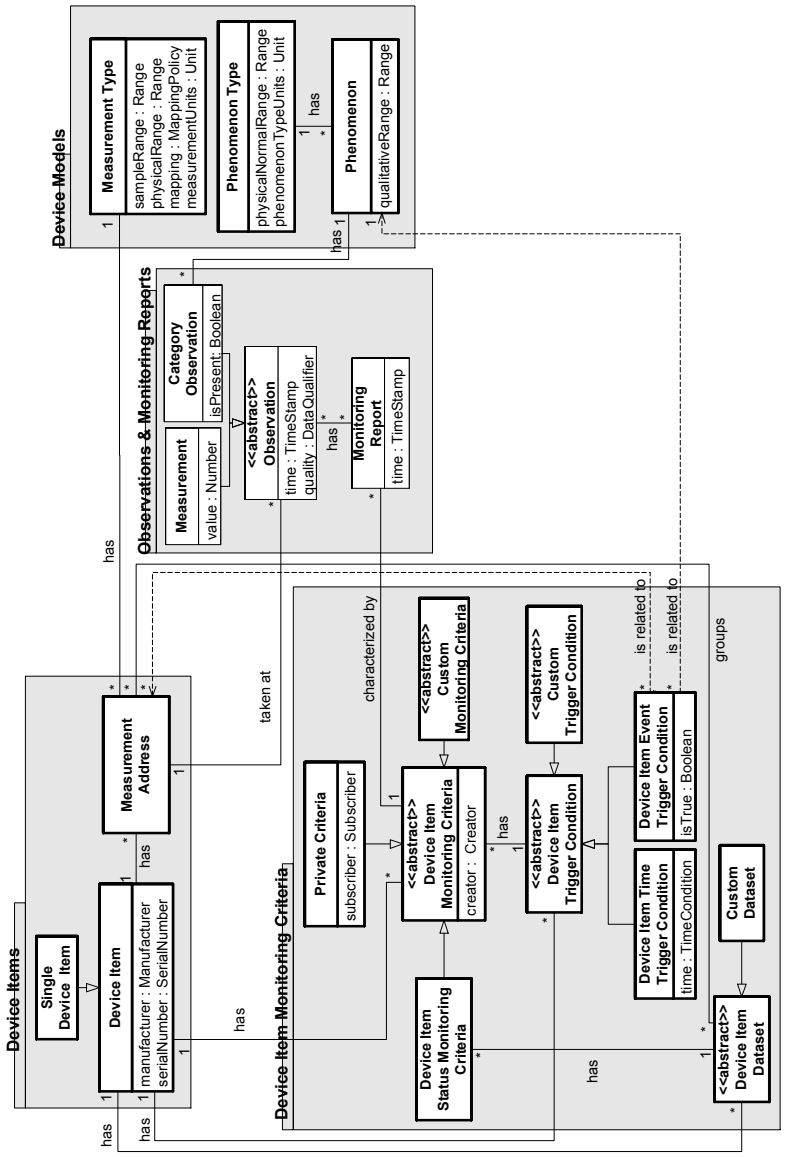

Figure 21 ODAS Conceptual Model

We summarize the information that we draw out from the comparisons, shown in Table 1, in the following points:

(i) *Lack of Models*. Some systems (such as OPC, IVI and ODAS) do not support the notion of *model*. This implies that many of the concepts regarding models (*Device Model* from the *Device Models* package; all the concepts defined in the *Device Model Monitoring Criteria* package; *Predefined Monitoring Criteria*, *Predefined Trigger Condition* and *Predefined Dataset* from the *Device Item Monitoring Criteria* package) do not exist either. This means that information (such as instances of *Measurement Point*, *Measurement Type*, P*henomenon Type* and *Phenomenon*) that is common to all the

instances of *Device Item* associated with the same instance of *Device Model* must be repeated for each instance. This duplication of *knowledge-level* data is rather inefficient and error prone.

Table 1 Concept Comparison of DAS Standards

Generic DAS Concept	OPC (v2)	IVI	ODAS
Device Models			
Single Device Model	×	✓	×
Complex Device Model	×	✓	×
Functional Device Model	×	✓	×
Measurement Point	×	✓	✓
Measurement Type	✓	✓	✓
Phenomenon Type	✓	✓	✓
Phenomenon	✓	✓	✓
Device Items			
Single Device Item	✓	✓	✓
Complex Device Item	✓	✓	×
Device Address	✓	✓	×
Measurement Address	✓	✓	✓
Device Model Monitoring Criteria			
Device Model Composition Monitoring Criteria	×	✓	×
Device Model Event Monitoring Criteria	×	✓	×
Device Model Status Monitoring Criteria	×	✓	×
Device Model Time Trigger Condition	×	✓	×
Device Model Event Trigger Condition	×	✓	×
Device Model Dataset	×	✓	×
Device Item Monitoring Criteria			
Public Criteria	✓	✓	×
Private Criteria	✓	×	✓
Device Item Composition Monitoring Criteria	×	✓	×
Device Item Event Monitoring Criteria	✓	✓	×
Device Item Status Monitoring Criteria	✓	✓	✓
Predefined Monitoring Criteria	×	✓	×
Device Item Time Trigger Condition	✓	✓	×
Device Item Event Trigger Condition	✓	✓	✓
Predefined Trigger Condition	×	✓	×
Custom Dataset	✓	✓	✓
Predefined Dataset	×	✓	×
Observations & Monitoring Reports			
Measurement	✓	✓	✓
Category Observation	✓	✓	✓
Monitoring Report	✓	✓	✓

(ii) *Lack of Composition*. Some systems (such as ODAS) do not support the notion of *composition*. This means, they do not deal with complex devices composed of other (single or complex) devices, neither in the domain of models nor in the domain of device items. They only deal with single devices. This is typically the case of a DAS that acquires data from a single device item.

(iii) *Plug&Play*. Some systems (such IVI) support the notion of *Plug&Play*. In these systems it is possible to define *predefined*, or *custom*, *composition monitoring criteria*. Then, a client can subscribe to receive notifications when changes on the composition of certain devices happen. Systems that do not support the notion of *Plug&Play* (such as OPC and ODAS) do not have concepts regarding *Composition Monitoring Criteria* (*Device Model Composition Monitoring Criteria* from the *Device Model Monitoring*

Criteria package and *Device Item Composition Monitoring Criteria* from the *Device Item Monitoring Criteria* package).

(iv) *Public Criteria versus Private Criteria*. Some systems (such as IVI) allow clients to define only *public criteria*. Some other systems (such as ODAS) allow clients to define only *private criteria*. There are also systems (such as OPC) that allow clients to define both *public criteria* and *private criteria*.

5.3 Design of a New System

A conceptual model of a generic system can be applied in the analysis phase of a particular system to better understand the problem and to easier specify the best solution, depending on the specific requirements of a particular system. A conceptual model of a generic system can also be used as a starting point for the design of a particular system. As a result, the use of a conceptual model of a generic system will save time and reduce the costs of the development of a particular system. In order to validate this hypothesis we developed a DAS for railway equipment based on our generic DAS conceptual model. In this section we summarize the major results and conclusions from the development of this DAS.

Development of a DAS for Railway Equipment. The main objective of this development was to validate our generic DAS conceptual model by means of an example. The development of this DAS gives also developers a case study on the development of a particular system based on a conceptual model of a generic system. Additionally, as part of the development of the DAS for railway equipment, we implemented a generic library of concepts, independent from the context of railway equipment, that can be reused and/or extended for the implementation of another DAS based on our generic DAS conceptual model. We analyzed our generic DAS conceptual model to obtain a DAS conceptual model specialized in the context of railway equipment, shown in Figure 22, going from the packages with fewer dependencies to the packages that have more dependencies:

1. We analyzed the *Device Models* package in the context of railway equipment. The result of this analysis was a partial conceptual model of the system that consists of a three-level hierarchy of device models: at the top level there are train models that are composed of vehicle models that are composed of equipment models. *Railway Equipment Model* is a specialization of *Device Model*; *Train Model* and *Vehicle Model* are specializations of *Complex Device Model*; and *Equipment Model* is a specialization of *Single Device Model* and *Functional Device Model*.

2. We analyzed the *Device Items* package in the context of railway equipment. From this analysis we included, in the conceptual model of the system, a three-level hierarchy of items: at the top level there are train items that are composed of vehicle items that are composed of equipment items, each of these items being characterized by its corresponding train, vehicle or equipment model. *Railway Equipment Item* is a specialization of *Device Item*; *Train Item* and *Vehicle Item* are specializations of *Complex Device Item*; and *Equipment Item* is a specialization of *Single Device Item*.

3. We analyzed the *Device Model Monitoring Criteria* package in the context of railway equipment. One of the requirements of the system was to be able to use datasets predefined in a model of train to record status monitoring reports of particular train items. Therefore, we included the concept of *Train Model Dataset*, which is a specialization of *Model Dataset* that groups measurement points corresponding to a train model or to one of its vehicle models or equipment models.

4. We analyzed the *Device Item Monitoring Criteria* package in the context of railway equipment. The requirements of the system were: (i) to enable the recording of status monitoring reports and event monitoring reports corresponding to a train item, such reports being triggered either by a time trigger condition or by an event trigger condition; (ii) to enable the definition of both *private* and *public* monitoring criteria; and (iii) to enable the definition of custom and predefined train datasets. Therefore, we introduced the concepts of: *Train Item Monitoring Criteria*, *Train Item Public Criteria*, *Train Item Private Criteria*, *Train Item Status Monitoring Criteria* and *Train Item Event Monitoring Criteria*, which are specializations of *Device Item Monitoring Criteria*, *Public Criteria*, *Private Criteria*, *Device Item Status Monitoring Criteria* and *Device Item Event Monitoring Criteria*, respectively; *Train Item Trigger Condition*, *Train Item Time Trigger Condition* and *Train Item Event Trigger Condition*, which are

specializations of *Device Item Trigger Condition*, *Device Item Time Trigger Condition* and *Device Item Event Trigger Condition*, respectively; and *Train Item Dataset*, *Train Custom Dataset* and *Train Predefined Dataset*, which are specializations of *Device Item Dataset*, *Custom Dataset* and *Predefined Dataset*, respectively.

5. Finally, we analyzed the *Observations and Monitoring Reports* package in the context of railway equipment. The requirements of the system were that both measurements and category observations could be recorded. Additionally, status and event monitoring reports could also be recorded. Therefore, we introduced the concepts of *Observation*, *Measurement*, *Category Observation* and *Monitoring Report* to the conceptual model of the system.

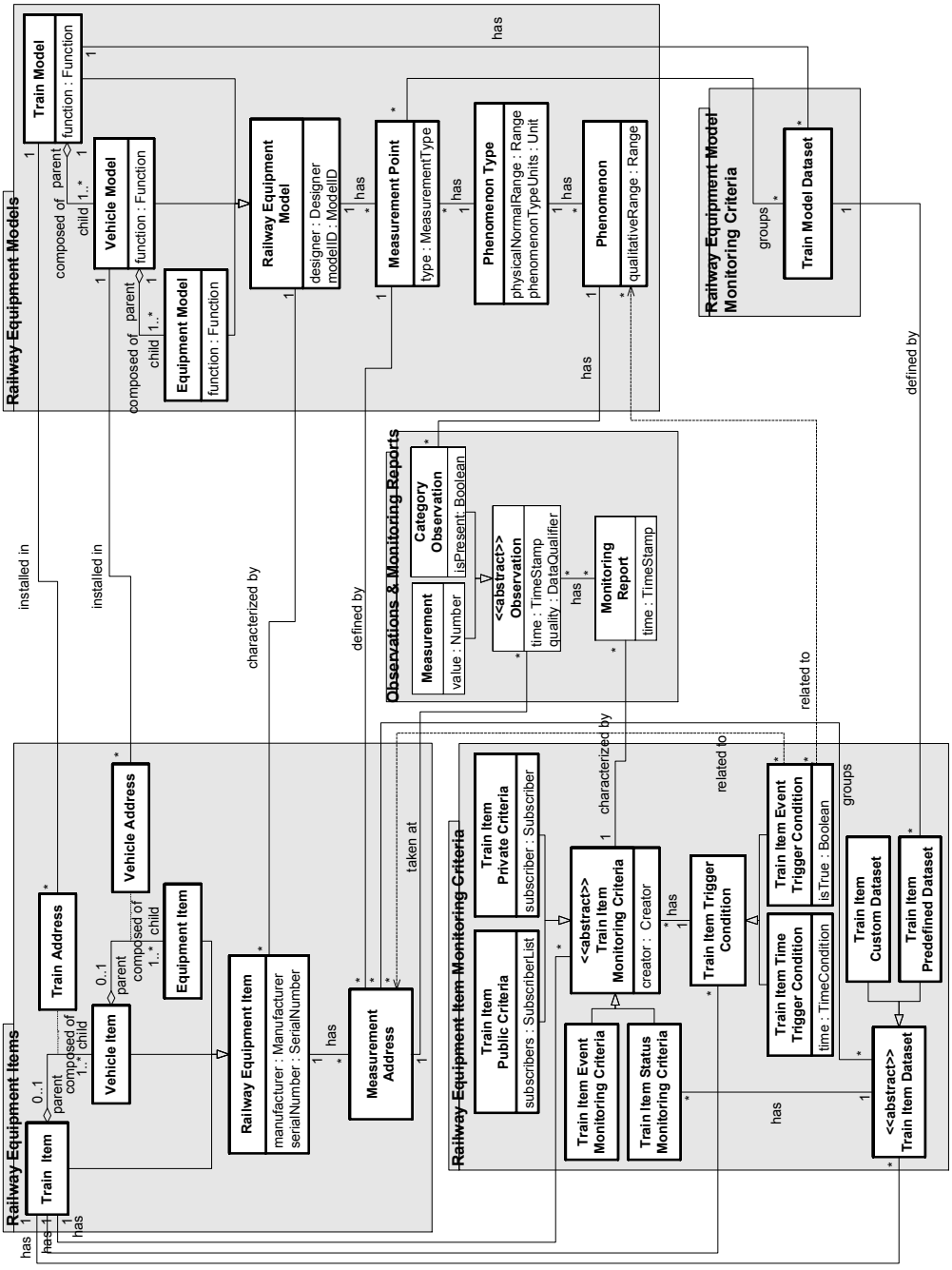

Figure 22 Railway Equipment DAS Conceptual Model

Our generic DAS conceptual model had a significant role in the design of the DAS railway equipment. We used this model to better organize the data that will be used by the components of the system. The conceptual model also made it easier the design of some of these components of the system. As a result, we had an implementation where a significant amount of the designed classes represent concepts specified in our generic DAS conceptual model. The development of this particular DAS demonstrated, by means of an industrial example, the usefulness of a conceptual model of a generic system for the development of a particular system.

6. Conclusions

In this paper, we described a conceptual model of a generic DAS. This model gives DAS developers an abstraction of DASs; it enables them to compare existing products and standards; and it provides the DAS developers that aim to develop a specific DAS with a starting point for the design of a specific DAS. We have found that a conceptual model of a generic system has many advantages. We propose patterns and techniques that are useful for the development of conceptual models of generic systems.

The most direct application of a conceptual model of a generic system is for the writing of a RFP for a new standard. Another potential application of a conceptual model of a generic system is for the evaluation of existing systems, standards or RFP responses. We illustrated this by using our generic DAS conceptual model to compare the different DAS standards. Finally, a conceptual model of a generic system can be applied in the development of a particular system. This will significantly reduce the development costs of a specific system. We illustrated this by means of an example of development of a DAS for railway equipment based on our generic DAS conceptual model. This development demonstrates, by means of an industrial example, the usefulness of a conceptual model of a generic system for the development of a particular system.

A conceptual model only specifies the static concepts of a system. We are currently working on the specification, using role-based use case modeling, of the dynamic behavior of a generic DAS. The role-based use case and conceptual models of a generic DAS will provide a complete specification of a generic DAS.

References

[Appleton, 1997] Appleton, B., 1997, "*Patterns and Software: Essential Concepts and Terminology*", November 20, http://www.enteract.com/~bradapp/docs/patterns-intro.html.

[Boman et al., 1997] Boman, M., J. A. Bubenko Jr., P. Johannesson, and B. Wangler, 1997, "*Conceptual Modelling*", Prentice Hall.

[Carzaniga et al., 1997] Carzaniga, A., G. P. Picco, and G. Vigna, 1997, "*Designing Distributed Applications with Mobile Code Pararadigms*" presented at 19th International Conference on Software Engineering (ICSE'97), Boston, Massachusetts, USA.

[D'Souza and Wills, 1999] D'Souza, D. F. and A. C. Wills, 1999, "*Objects, Components, and Frameworks with UML - The Catalysis Approach*", Addison-Wesley.

[Fabri et al., 1999] Fabri, A., T. Nieva, and P. Umiliacchi, 1999, "*Use of the Internet for Remote Train Monitoring and Control: the ROSIN Project*" presented at Rail Technology '99, London, UK, http://icawww.epfl.ch/nieva/thesis/Conferences/RailTech99/article/RailTech99.PDF.

[Fowler, 1997] Fowler, M., 1997, "*Analysis Patterns: Reusable Object Models*", Addison-Wesley, http://www2.awl.com/cseng/titles/0-201-89542-0/apsupp/index.htm.

[Gamma et al., 1995] Gamma, E., R. Helm, R. Johnson, and J. Vlissides, 1995, "*Design Patterns - Elements of Reusable Object-Oriented Software*", Addison-Wesley.

[Itschner et al., 1998] Itschner, R., C. Pommerell, and M. Rutishauser, 1998, "*GLASS: Remote Monitoring of Embedded Systems in Power Engineering*" in IEEE Internet Computing, vol 2.

[IVI Foundation, 1997] IVI Foundation, 1997, "*Interchangeable Virtual Instruments Standard*", http://www.ivifoundation.org/.

[Jini User Group, 2000] Jini User Group, 2000, "*Jini Home Page*", http://www.jini.org.

[Mansouri-Samani and Sloman (1994)] Mansouri-Samani, M. and M. Sloman, 1994, "*Monitoring Distributed Systems*" in Network and Distributed Systems Management, Addisson-Wesley.

[Motschnig-Pitrik and Kaasboll, 1999] Motschnig-Pitrik, R. and J. Kaasboll, 1999, "*Part-Whole Relationship Categories and Their Application in Object-Oriented Analysis*" in IEEE Transactions on Knowledge and Data Engineering vol. 11, pp. 779-797.

[MSDN, 1994] MSDN, 1994, "*Microsoft Windows and the Plug and Play Framework Architecture*", http://msdn.microsoft.com/library/backgrnd/html/msdn_pnp.htm.

[Nieva, 1999] Nieva, T., 1999, "*Automatic Configuration for Remote Diagnosis and Monitoring of Railway Equipment*" presented at IASTED International Conference - Applied Informatics, Innsbruck, Austria, http://icawww.epfl.ch/nieva/thesis/Conferences/ai99/article/ai99.pdf.

[ODAA, 1998] ODAA, 1998, "*Open Data Acquisition Standard*", http://www.opendaq.org/.

[Olken et al., 1998] Olken, F., H. A. Jacobsen, C. McParland, M. A. Piette, and M. F. Anderson, 1998, "*Objects lessons learned from a distributed system for remote building monitoring and operation*" presented at Conference on Object-oriented Programming, Systems, Languages and Applications, Vancouver, Canada, http://www.lbl.gov/~olken/rbo/rbo.html.

[OMG, 1999] OMG, 1999 "*Data Acquisition from Industrial Systems (DAIS)*", Request for Proposal (RFP), OMG Document: dtc/99-01-02, January 15, http://www.omg.org/techprocess/meetings/schedule/Data_Acquisition_RFP.html.

[OPC Foundation, 1997] OPC Foundation, 1997, "*OLE for Process and Control Standard*", http://www.opcfoundation.org.

[OPC Foundation, 1999] OPC Foundation, 1999 "*OPC - OLE for Process Control, Data Access Automation Interface Standard*", Version 2.02, February 4, http://www.opcfoundation.org/specs.asp.

[Rumbaugh et al., 1999] Rumbaugh, J., I. Jacobson, and G. Booch, 1999, "*The Unified Modelling Language Reference Manual*", Addison Wesley, http://www.rational.com, http://www.omg.org.

[UPnP Forum, 2000] UPnP Forum, 2000, "*Universal Plug and Play Home Page*", http://www.upnp.org/.

[Wireman, 1994] Wireman, T., 1994, "*Computerized Maintenance Management Systems*", Industrial Press, Inc.